Verbal Plan Explanations for Hybrid Planning

Julien Bidot[1], Susanne Biundo[1], Tobias Heinroth[2],
Wolfgang Minker[2], Florian Nothdurft[2], Bernd Schattenberg[1]

[1]Institute of Artificial Intelligence,
[2]Institute of Information Technology,
(both) Ulm University

1 Introduction

State-of-the-art AI planning systems are able to generate complex plans thanks to their efficient reasoning engines. In a large number of application domains, the plans are automatically executed by systems such as autonomous robots. In this context, it is not necessary to make these automated systems understand what they are actually doing during execution and why they are doing that. In other words, these systems do not need to understand the underlying semantics of the plans they execute and how these plans have been generated.

However, there are a significant number of key application domains, such as disaster relief mission support or project planning, where plans are supposed to be executed by a human user who is not necessarily a planning expert, an application expert, or both. In addition, for real-world applications, the plans and the plan generation are often complex. In order to unlock a part of the application potential of the AI planning technology, it is necessary that the user trusts the technology (Glass et al. 2008, pp. 12-18). Increasing trust in AI planning systems requires the design and implementation of user-friendly interfaces and the development of plan explanation methods that allow for taking into consideration the human user's queries related to some components of the plan about their meaning and relevance for the plan and giving back the appropriate information that answers these queries.

The verbal communication by speech constitutes the most natural form of communication for humans. By means of natural language dialogs in this work, we focus on the *explanation of plans* that are generated by a refinement-based planning system. Contrary to most approaches presented in the literature that try to provide explanations when backtracking occurs in failure situations during search, we assume in this work that the plans for which explanations are looked for are consistent. We present a *domain-independent* approach to enabling verbal human queries and producing verbal plan explanations.

The next section serves as an introduction to AI planning. Section 3 introduces plan explanations. Section 4 presents the formal framework for refinement-based planning that responds to the requirements imposed by the generation of plan explanations. In Section 5, we detail how to use refinement-based planning to explain plans to a human user. Section 6 describes our prototype system that allows for verbal user queries and the generation of verbal plan explanations. In Section 7, some related work is presented. We conclude and propose some future work in Section 8.

2 AI Planning

In Artificial Intelligence (AI), the *classical planning problem* consists in a set of operators, one initial state, and one goal state. The instance of an operator is called a task. A state is a set of positive literals. The world state can evolve; e.g., it changes when a task is executed, since every executed task has usually an effect on the current world state. The objective is to select and organize tasks in time that allow one to attain the goal state from the initial state. These tasks and ordering constraints constitute a *plan*. Tasks can be executed in a world state, only if some preconditions hold in this state. Each task is thus associated with preconditions and effects. A plan consists of tasks and temporal and causal relationships between these tasks. In a recent book, Ghallab, Nau, and Traverso (2004) present a survey about AI planning.

A planning problem is a complex problem to solve, since it is highly combinatorial: a large number of tasks to select and a huge number of conflicts that appear between tasks. Generated plans tend to be confusing, since they contain a large number of tasks and there are possibly a large number of ordering constraints between these tasks.

3 Plan Explanations

The general problem of designing and implementing interfaces between complex computer-based systems and humans appears in practice, when these two entities work together. In particular, if a plan is passed down to a human user in a form that only a machine can interpret, the user cannot find out the role and the meaning of the plan's tasks. In addition, if the plan contains a task that the user has to execute, the execution of the whole plan is thereby jeopardized. Here, the trust of the user in AI planning systems plays an important role. The approach we present to increase trust in these systems consists in providing the user with semantic and context-dependent information as well as logical information about plans. Depending on the nature of this information, we distinguish between *substantial knowledge* and *executive knowledge*.

3.1 Substantial Knowledge

A prerequisite for the understanding of a plan is the domain-specific *substantial knowledge*. During the implementation of the plan, the information about the involved physical systems and functions is notably important for users who are not application experts.

```
<domainModel name="satellite2">
  <relationDeclaration name="on_board" type="rigid">
    <utterance>var1 is on board of var2</utterance>
    <sortExpression name="instrument"/>
    <sortExpression name="satellite"/>
  </relationDeclaration> . . .
  <taskDeclaration name="calibrate" type="primitive">
      <utterance>calibrate var2 on var1 in var3</utterance>
      <documentation>Calibrating an instrument is needed for
      taking a sharp image. The calibration is executed on a
      known object. </documentation>
      <varDeclaration name="c_s" sort="satellite"/>
      <varDeclaration name="c_i" sort="instrument"/>
      <varDeclaration name="c_d" sort="calib_direction"/>
       . . .
  </taskDeclaration>
<taskDeclaration name="activate_instrument" type="complex">
    <utterance>activate instrument on satellite</utterance>
    <documentation>
      Activating an instrument optionally includes turning off
      other instruments on the same satellite in order
      to provide energy, followed by turning on the instrument.
    </documentation>
    <varDeclaration name="ai_s" sort="satellite"/>
    <varDeclaration name="ai_i" sort="instrument"/>
  </taskDeclaration>
```

Figure 1: Domain Model with Additional Tags for Substantial Knowledge

A classical domain model used for automated planning only contains the declaration of literals and tasks, but does not give any additional information about the use and role of these literals and tasks. Since the plan-explanation generation components of our approach are *domain-independent*, we need to extend the document describing the application domain for planning in order to make this additional information accessible to human users. In this way, we limit the effort necessary for adapting an existing planning domain for delivering explanations to human users. Figure 1shows an extract of a domain model in which additional tags are highlighted in brown. The domain model is enhanced with *documentation* tags that contain general descriptions of plan components, such as tasks and task parameters. For example, it is possible to explain to the human user that the role

of task "calibrate" in the satellite domain is to calibrate an instrument, which is needed for taking a sharp image. The domain is also enriched with *utterance* tags that give the human user some context-dependent descriptions of tasks; i.e., these descriptions depend on how task parameters are instanciated for a given plan. For example, for a particular task such as "calibrate(EutelSat,Thermograph2,Jupiter)" of the plan, it enables the human user to get to know that calibration direction Jupiter is used to calibrate instrument Thermograph2 of satellite EutelSat.

3.2 Executive Knowledge

The generation of plan explanations for a human user who is not a planning expert requires the analysis of the plan and the set of successful planning decisions that have led to the plan during search. The analysis is difficult to grasp by a human user because the logical dependencies between tasks of the plan are not necessary direct. In order to provide the logical description of a plan, the context-dependent information about its tasks is necessary but not sufficient, it is also necessary to understand the temporal and causal relationships between the tasks. This logical description of the plan relies on *executive knowledge*. The human users without planning expertise who are presented a plan would possibly ask for explanations about the ordering or temporal positions of tasks of the plan.

The generation of plan explanations based on executive knowledge has to be implemented in a generic way in order to enable a *domain-independent* use.

3.3 Requirements

In order to ensure a smooth and natural dialog between the human user and the plan-explanation generation engine, the plan-explanation generation engine has to interpret the human user's queries without ambiguity and quickly provide plan explanations that correspond to these queries. This requires a consistent representation of plans and plan components, and an easy access to them and to successful planning decisions; i.e., the plan components and the planning decisions have to be explicitly represented.

4 A Formal Framework for Refinement-Based Planning

Our approach to generating verbal plan explanations is based on a hybrid planning framework that integrates partial-order causal-link planning and hierarchical planning (Biundo and Schattenberg 2001, pp. 157-168). This framework uses an ADL-like representation of states and basic actions (*primitive tasks*). States, preconditions, and effects of tasks are specified through formulae of a fragment of first-order logic. *Abstract tasks* can be refined by so-called *decomposition methods*, which provide

task networks (*partial plans*) that describe how the corresponding task can be solved. Partial plans may contain abstract and primitive tasks. With that, hierarchies of tasks and associated methods can be used to encode the various ways to accomplish an abstract task.

A *domain model* $D = \langle T, M \rangle$ consists of a set of task schemata T and a set M of decomposition methods. A partial plan is a tuple $P = \langle TE, \prec, VC, CL \rangle$, where TE is a set of *task expressions* (plan steps) $te = l{:}\, t(\bar{\tau})$ with t being the task name and $\bar{\tau} = \tau_1, \tau_2, \ldots, \tau_n$ the task parameters; the label l is used to uniquely identify the steps of the plan. $\prec$ is a set of *ordering constraints* that impose a partial order on plan steps of TE. VC are *variable constraints* i.e. co-designation and non-co-designation constraints on task parameters. Moreover, VC contains sort restrictions that restrict further co-designations. CL are *causal links* and provide the usual means to establish and maintain causal relationships among the tasks in a partial plan. A causal link $\langle te_i, \phi, te_j \rangle$ indicates, that formula ϕ, which is an effect of task te_i, supports (a part of) the precondition of task te_j. A *planning problem* $\pi = \langle D, P_{\text{init}} \rangle$ consists of a domain model D and an initial task network P_{init}. The *solution* of a planning problem is obtained by transforming P_{init} stepwise into a partial plan P that meets the following *solution criteria*: (1) all preconditions of the tasks in P are safely supported by causal links; (2) the ordering and variable constraints of P are consistent; (3) all steps in P are primitive tasks.

Transforming partial plans into their refinements is done by using so-called *plan modifications*. Given a partial plan $P = \langle TE, \prec, VC, CL \rangle$ and domain model D, a plan modification is defined as $m = \langle E^{\oplus}, E^{\ominus} \rangle$, where $E^{\oplus}$ and $E^{\ominus}$ are disjoint sets of elementary additions and deletions of *plan components* over P and D. Consequently, all elements in $E^{\ominus}$ are from TE, $\prec$, VC, or CL, respectively, while $E^{\oplus}$ consists of new plan components. This generic definition makes the changes explicit that a modification imposes on a plan. Applying a modification $m = \langle E^{\oplus}, E^{\ominus} \rangle$ to a plan P returns a plan P' that is obtained from P by adding all components of $E^{\oplus}$ and removing those of $E^{\ominus}$. Hybrid planning distinguishes various classes of plan modifications such as task expansion, causal link insertion, and task insertion.

For a partial plan P that is a refinement of the initial task network of a given problem, but is not yet a solution, so-called *flaws* are used to make the violations of the above criteria explicit. Flaws list those plan components that constitute deficiencies of the partial plan. We distinguish various flaw classes including the ones for causal threats, unsupported preconditions of tasks, and inconsistencies of variable and ordering constraints.

It is obvious, that particular classes of modifications are appropriate to address particular classes of flaws while others are not. For example, the modifications of the class "causal link insertion" can address the flaws of the class "unsupported preconditions of tasks." A *modification trigger* function, which is used in the algo-

rithm presented by Schattenberg, Weigl, and Biundo (2005, pp. 258-272), relates flaws to modifications that are suitable to eliminate them.
The plan generation process works as follows: (1) the flaws of the current plan are collected; if no flaw is detected, the plan is a solution; (2) suitable modifications are generated using the modification trigger; if for some flaws no modification can be found, the plan is discarded (dead-end); (3) selected modifications are applied and generate further refinements of the plan; (4) the next candidate plan is selected and we proceed with (1).

This formal framework very well responds to the requirements imposed by the generation of verbal plan explanations, since plan components and plan decisions (i.e. plan modifications) are explicitly represented.

5 Plan Explanations for Refinement-Based Hybrid Planning

When considering a plan P generated by a planning system based on our formal framework, the retrieval of executive knowledge is quite obvious: we can easily access the task components of P, including task expressions, task parameters, and variable constraints.

However, the understanding of individual tasks of the plan is not sufficient to comprehend the whole plan, since its temporal and causal structure links the tasks. In other words, in order to grasp the whole plan, a human user needs to understand how the plan was generated with the help of executive knowledge. Our formal framework for refinement-based planning allows for an explicit representation of the search space explored during the plan generation process thanks to flaws and plan modifications, and it is possible to explore this search space backwards in order to search for the relevant flaws and plan modifications that explain the presence of particular components in the plan, such as tasks, causal links, and ordering constraints. The complexity of this analysis depends on the number of successful planning decisions and on the temporal and causal structure of the plan. In this work, we assume that a human user becomes a set of tasks that are partially ordered and asks for explanations that can justify the ordering of two tasks, or the temporal position of a particular task. For each user's query, the number of plan explanations has to be relatively small, otherwise the user is overwhelmed with information.

5.1 Explanations for the Ordering of Two Tasks

In our formal framework, the most important plan components to consider during the analysis of the search space for finding some plan explanations justifying the ordering of two tasks are *causal links* and *ordering constraints*.
A causal link expresses a direct dependency between two tasks and indicates in the same time the required ordering of them. The analysis accounts not only for direct

causal links but also for indirect causal links. The logical dependency between two tasks te_i and te_j is indeed not necessarily direct; e.g. there may be another task te_k that is directly related to te_i and te_j, respectively, via causal links.

The two tasks te_i and te_j can be directly ordered by a single ordering constraint or indirectly ordered by several ordering constraints if some tasks are placed between te_i and te_j. Explaining the presence of an ordering constraint between two tasks reverts to finding a set of flaws and plan modifications that are related to the insertion of this ordering constraint. There are two classes of plan modifications that are responsible for the insertion of an ordering constraint into a plan: "task expansion" and "ordering constraint insertion." The class "task expansion" is adequate to address the flaws of the following classes: "abstract task," "unsupported precondition," and "causal threat." The class "ordering constraint insertion" is appropriate to eliminate the flaws of class "causal threat." Explanations for inserting ordering constraints into a plan are indeed diverse: presence of an abstract task, an unsupported precondition, or a causal threat. For example, Figure 2 represents a linear plan and the task hierarchy associated it. The plan is composed of 5 primitive tasks. The 4 arrows represent ordering constraints between these tasks. During the plan generation process, the abstract task "do_observation" was expanded into two tasks: "activate_instrument" and "take_image." When considering the ordering of two tasks that were inserted into the plan by expanding an abstract task, the plan explanation for this ordering is the task expansion defined in the planning domain; e.g., the tasks "turn_to" and "calibrate" are ordered and were inserted into the plan in order to expand the abstract task "auto_calibrate."

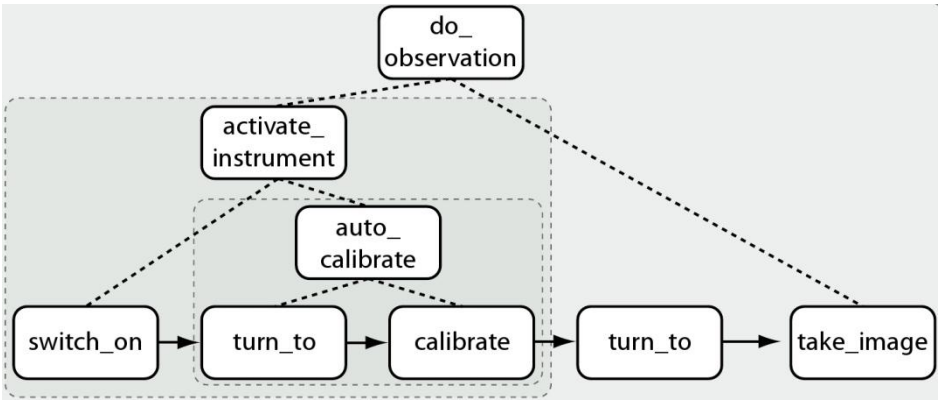

Figure 2: A Linear Plan and its Task Hierarchy for the Satellite Planning Domain

The order in which the classes of plan explanations are given to the user is predefined in this work. An ordering of two tasks which is due to the removal of a causal threat is of upmost priority. Of lower priority is the expansion of an abstract

task. Furthermore, the direct causal links are more important than the indirect causal links for plan explanations, since they express direct causal dependencies.

5.2 Explanations for the Temporal Position of a Single Task

While the analysis concentrates only on the relationships between two tasks for justifying the ordering of them, the whole plan has to be analyzed for generating plan explanations for the temporal position of a single task. In this context, it is also more complex to generate plan explanations, since the temporal position of this single task te_i is relative to other tasks. The analysis is composed of two parts: (1) the identification of the direct predecessor tasks and the direct successor tasks of te_i (i.e. the direct neighbor tasks of te_i); (2) the search for some plan explanations that justify the ordering of te_i and its direct neighbor tasks.

The identification of the direct neighbor tasks of te_i can be done by analyzing the ordering constraints between all the tasks of the plan.

As for the ordering of two tasks, there are three types of explanations for inserting ordering constraints between te_i and its neighbor tasks: presence of an abstract task, an unsupported precondition, and a causal threat. The removals of causal threats with respect to the task te_i during the plan generation process are very important, since these ordering constraints are often inserted between te_i and its direct neighbor tasks. By analogy, the direct causal links of te_i can be connected to the direct neighbor tasks of te_i. If te_i and at least one of its direct neighbor tasks were inserted into the plan by expanding an abstract task, then the plan explanation for the ordering of these tasks is the task expansion defined in the planning domain.

6 Experimental System

For some types of information, the graphical modality is adequate; the complete task hierarchy associated with a plan would be difficult to express verbally. For some other types of information, speech is better suited: semantically rich data, such as the description of a machine or the functions thereof can often be better disclosed via spoken language. Moreover, specific plan explanations can be given quickly and concisely via spoken language. In order to implement our approach to generating verbal plan explanations, we have created a prototype system that allows for speech interaction with a human user. The basic components of the prototype system are shown in Figure 3.

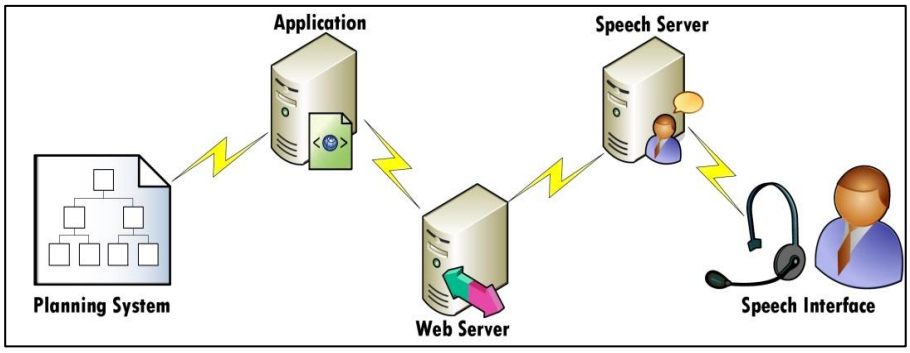

Figure 3: Prototype Architecture

The planning system we have used for our experiments is PANDA, which integrates hierarchical planning and partial-order causal-link planning. Note, that we did not modify PANDA for realizing this prototype system, which means that other planning systems could be possibly used instead of PANDA. When a plan is generated, the information coming from the plan generation process is preprocessed and stored in the component "application" to speed up the plan-explanation generation process: XML data which contains the task hierarchy corresponding to the plan is created. During the preprocessing phase, a partial representation of the plan is stored in a further XML file in which every task of this plan is associated with some substantial knowledge. This document, which is the basis for the interactions with the human user, represents the tasks and the ordering constraints of the plan but does not contain any information about causal links. It provides the basis for interaction, since it is possible to create adequate grammars stored in VoiceXML data from this. This procedure is based on transforming the XML documents with the help of a fitting XSL stylesheet into VoiceXML documents. The created documents are then interpreted by the speech server that accesses them via a web server. This spoken language dialog system (SLDS) also integrates a speech interface to provide the dialogue to the user.

The human user communicates with the prototype system via a Voice over Internet Protocol software. Once the human user gets a plan, the SLDS gives four possible alternatives to him/her: to describe the plan, to describe some components of the plan, to explain the ordering between two tasks, or to explain the temporal position of a single task of the plan. These dialogs are generated dynamically, since the selection of a plan component depends on the plan under consideration.

For example, the user's query for explanation of an ordering between two tasks is transmitted to the component "application" via a web server. The component "application" analyzes the executive and substantial knowledge and generates then

plan explanations. These explanations are then transformed into a VoiceXML fragment, the content of which is finally read to the human user.

7 Related Work

Most existing work in AI planning confines itself to representing visually abstract plan concepts. The plan-generation process and the temporal ordering of tasks are thereby not adequately described for people who are neither planning experts nor application experts. In most cases, the semantic and the provenance of plan components are cryptic for non-expert users. The most common representation form of a plan is thereby a graph, which allows for visualizing the temporal ordering of its tasks.

Vrakas and Vlahavas (2005, pp. 975-998) use ViTA Plan, a visual tool for adaptive planning that is able to display a plan in the form of a graph in which the nodes represent tasks and the arrows represent causal links between tasks. Agosta and Wilkins (1996, pp. 6-8) and Kundu et al. (2002) propose a graphical representation of a plan, but the tools they present allow for the visualization of the whole plan without explaining the underlying causal and temporal structure of the plan to end users.

Contrary to our work, the term "explanation" in the AI planning literature is commonly associated with the set of elements that cause backtracking in failure situations during search. Zimmerman and Kambhampati (1999, pp. 605-611) use the term *pilot explanation* to designate the failures that occur in the previous search levels of the graph developed using the Graphplan algorithm (Blum and Furst 1997, pp. 281-300). The idea is to maintain the pilot explanation structure capturing the failures encountered at previous levels of search, using it to guide the search at the next levels.

8 Conclusion and Future Work

In this work, we presented a domain-independent approach to generating verbal plan explanations. This approach is based on a formal framework for refinement-based hybrid planning. We have developed a prototype system that can address different types of verbal human queries: given a plan, he/she can ask not only for semantic or context-specific information about plan components, but also for justifications about the ordering of two tasks or the temporal position of a single task. This work unlocks a part of the application potential of the AI planning technology, as it instills trust in the user of the technology thanks to the support for plan explanations.

The unimodal verbal explanations of a plan are not sufficient. The human user will not always be able to deal with the large amount of information associated with the explanations. But a large number of explanations with a graphical repre-

sentation can also be quickly confusing. Choosing the adequate modality used to represent a specific plan explanation seems to be a key issue in this context. In other words, multimodal approaches to generating plan explanations are promising future work.
The quality of plan explanations could be improved by modifying planning domains and the component "application" of the prototype system. Associating weights to causal links depending on how important they are in the application domain could allow for filtering plan explanations. Another improvement would be to take into account the user's profile and adapt the plan explanation generation to it.

Acknowledgement

The research leading to these results has received funding from the European Community's 7th Framework Programme (FP7/2007-2013) under grant agreement n° 216837 and from the Transregional Collaborative Research Centre SFB/TRR 62 "Companion-Technology for Cognitive Technical Systems" funded by the German Research Foundation (DFG).

References

Agosta JM, Wilkins DE (1996) Using SIPE-2 to plan emergency response to marine oil spills. IEEE Expert 11(6):6-8.

Biundo S, Schattenberg B (2001) From abstract crisis to concrete relief—a preliminary report on combining abstraction and HTN planning. In: Cesta A, Borrajo D (eds.) Proceedings of the 6th European conference on planning, pp.157-168.

Blum A, Furst M (1997) Fast planning through planning graph analysis. Artificial Intelligence 90:281-300.

Ghallab M, Nau DS, Traverso P (2004) Automated planning: theory and practice. Morgan Kaufmann.

Glass A, McGuinness DL, Pinheiro da Silva P, Wolverton M (2008) Trustable task processing systems. Roth-Berghofer T, Richter MM (eds.) Künstliche Intelligenz Zeitschrift 2/08, Special Issue on Explanation:12-18.

Kundu K, Sessions C, desJardins M, Rheingans P (2002) Three-dimensional visualization of hierarchical task network plans. In: Proceedings of the 3rd international NASA workshop on planning and scheduling for space.

Schattenberg B, Weigl A, Biundo S (2005) Hybrid planning using flexible strategies. In: Furbach U (ed.) Proceedings of KI 2005. LNAI, vol. 3698, pp. 258-272. Springer, Heidelberg. doi: 10.1007/978-3-540-85845-4_21.

Vrakas D, Vlahavas I (2005) A visualization environment for planning. International journal on artificial intelligence tools 14(6):975-998.

Zimmerman T, Kambhampati S (1999) Exploiting symmetry in the planning graph via explanation-guided search. In: Proceedings of the 16th national conference on artificial intelligence and 11th conference on innovative applications of artificial intelligence, pp. 605-611.